AlexNet：ImageNet Classification with Deep Convolutional Neural Networks 论文学习笔记

一　简介

AlexNet 是 Alex Krizhevsky、Ilya Sutskever 和 **Geoffrey Hinton** 创造了一个“大型的深度卷积神经网络”，赢得了 2010 和 2012 ILSVRC（2012 年 ImageNet 大规模视觉识别挑战赛）。2012 年是 CNN 首次实现 Top 5 误差率 15.4%的一年（Top 5 误差率是指给定一张图像，其标签不在模型认为最有可能的 5 个结果中的几率），当时的第二名误差率为 26.2%。我们可以看出性能提升相当大。AlexNet 也是深度学习和神经网络的重新崛起转折点。正是由于 AlexNet 在 ImageNet 竞赛中夺冠，深度学习正是进入学术界的视野。

二　涉及技术

2.1　ReLU 激活函数

在 ReLU 激活函数未提出之前，神经网络的激活函数主要为 Sigmoid 函数和 Tanh 函数。这里两个函数的表达式如下：

$$Sigmoid(x)=\frac{1}{1+e^{-x}}$$

$$\begin{aligned}Tanh(x)&=\frac{1-e^{-2x}}{1+e^{-2x}}=\frac{2}{1+e^{-2x}}-1\\&=2Sigmoid(2x)-1\end{aligned}$$

两个函数的导数为：

$$Sigmoid'(x)=Sigmoid(x)\left[1-Sigmoid(x)\right]$$

$$Tanh'(x)=1\text{-}Tanh(x)^2$$

这两个函数的图像如图 2.1 所示。显然 Sigmoid 函数和 Tanh 的求导是极其方便的。Sigmoid 函数的值域为$(0,1)$，图像关于 0.5 中心对称；Tanh 函数值域为$(-1,1)$，图像关于 0 中心对称，有利于相关数值的特殊处理。无论是 Sigmoid 函数还是 Tanh 函数，两者在实数范围内连续可导，优化稳定。同时当$x\to\infty$时，函数区域平缓，导数趋向于 0，因此 Sigmoid 和 Tanh 函数都是饱和函数，容易在训练后期出现梯度消失，导致训练后期出现梯度消失问题。但是从图 2.1 也可以看出，Tanh 函数比 Sigmoid 函数收敛更快。

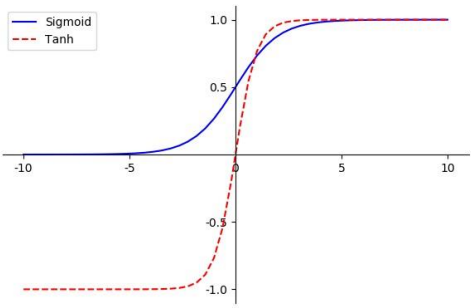

图 2.1　Sigmoid 函数和 Tanh 函数图像

下面给出饱和函数的定义。设函数为$\phi(x)$，若满足如下等式：

$$\lim_{x\to+\infty}\phi'(x)=0$$

则称$\phi(x)$为右饱和。同理，如满足等式：

$$\lim_{x\to-\infty}\phi'(x)=0$$

则称$\phi(x)$为左饱和。那么，$\phi(x)$为饱和的充要条件为：$\phi(x)$既是左饱和又是右饱和。

对任意的x，如果存在常数c，当$x>c$时，恒有$\phi'(x)=0$，则称$\phi(x)$是右硬饱和，当$x<c$时，恒有$\phi'(x)=0$，则称$\phi(x)$为左硬饱和。$\phi(x)$为硬饱和的充要条件为：$\phi(x)$既是左硬饱和又是右硬饱和。$\phi(x)$为软饱和的充要条件为：只有在极限状态下$\phi'(x)=0$。

在 AlexNet 论文当中，提出了利用非饱和非线性的 ReLU 函数作为激活函数。ReLU 函数的函数表达式及其导数如下：

$$ReLU(x)=\begin{cases}x & x\geq 0\\ 0 & x<0\end{cases}$$

$$ReLU'(x)=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases}$$

显然相比于 Sigmoid 函数和 Tanh 函数，ReLU 函数及其导数没有复杂的指数运算与乘除运算，减少了计算负担。ReLU 函数图像如图 2.2 所示。

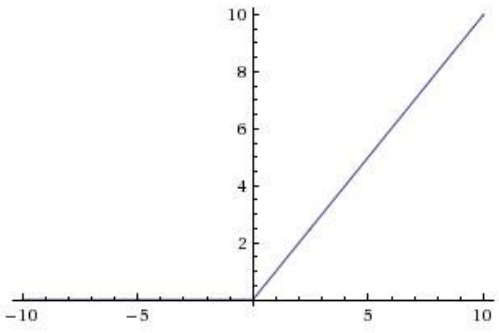

图 2.4　ReLU 函数图像

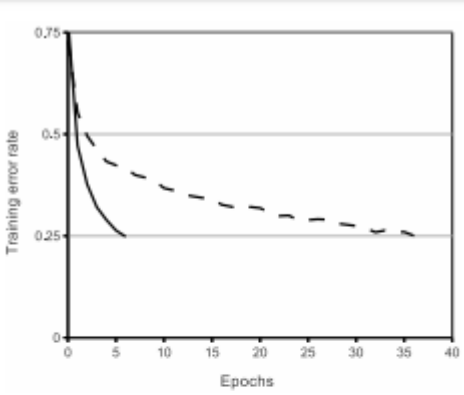

Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

图 2.3　论文实验截图

在论文中也指出，使用 ReLU 的深度卷积神经网络训练速度比同样情况下使用 Tanh 单元的速度快好几倍。图 2.3 表示使用特定的四层卷积网络在数据集 CIFAR-10 上达到 25%错误率所需的迭代次数。其中实线代表 ReLU，虚线代表 Tanh。图 2.3 表明如果使用传统的饱和激活函数不可能在大规模神经网络中利用。

2.2　多 GPU 并行训练

在该论文中采用 2 个 NVIDIA GTX 580 GPU 进行训练网络。单个 NVIDIA GTX 580 GPU 只有 3GB 的内存，这限制了可以在其上训练的网络的最大尺寸。在 LSVRC2010 数据集共有 120 万个训练样本。这 120 万个训练样例过于庞大，也无法放在一个 GPU 上的网络。因此，论文中将 AlexNet 网络分布在两个 GPU 上。当前的 GPU 特别适合跨 GPU 并行化，因为它们能够直接读取和写入彼此的内存，而无需通过主机内存实现并行化。

具体方案为：将一半内核（或神经元）放在每个 GPU 上，还有一个额外的技巧：GPU 只在某些层中进行通信。这意味着，例如，第 3 层的内核从第 2 层中的所有内核映射获取输入。但是，第 4 层中的内核仅从位于同一 GPU 上的第 3 层中的那些内核映射获取输入。

2.3　局部响应归一化(LRN)

ReLU 函数不需要对输入进行归一化来防止饱和。只要训练样本产生一个正输入给一个 ReLU 函数，那么在那个神经元中学习就会开始了。但是，论文中提出了局部响应归一化对神经元输入进行了改进。我们将坐标为 (x, y) 的像素在第 i 个核函数进行卷积操作之后，并利用 ReLU 函数的非线性进行激活的神经元记作 $a^i_{x,y}$，$b^i_{x,y}$ 为 $a^i_{x,y}$ 对应的局部响应标准化的神

经元。$b_{x,y}^{i}$ 的计算公式如下：

$$b_{x,y}^{i}=\frac{a_{x,y}^{i}}{\left[k+\alpha\sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)}\left(a_{x,y}^{i}\right)^{2}\right]^{\beta}}$$

其中，n 代表与 $a_{x,y}^{i}$ 在同一坐标上最近相邻核映射个数，N 为该层核函数的深度。k,α,β 为超参数。Tensorflow 中 LRN 的实现有两个接口：tf.nn.local_response_normalization 和 tf.nn.lrn。k 对应于函数中 bias；α 对应于函数中 alpha；$\frac{n}{2}$ 对应于函数中 deepth_radius；β 对应于函数中 belta。

由于内核映射的顺序是任意的，在训练前就已经决定好了。那么这种局部响应归一化实现了横向抑制，使得根据利用不同核计算得到的神经元输出之间产生了竞争。原始输出较大的值变得更大，而较小的值变得更小即实现了抑制，在这样的机制下提高了模型泛化能力。

在 AlexNet 中，这项技术技术在全连接层之前的所有层都是使用。其中参数设置全部一样：$k=2,n=5,\alpha=10^{-4},\beta=0.75$。

2.4 重叠池化

在 CNN 中，池化层总结了同一个核函数下相邻神经元的输出。传统的，相邻池化单元的总结不重叠。为了更精确，一个池化层可以被认为是由相邻 S 个像素的池化网格所组成，每个总结是池化单元中心的邻近 $z\times z$ 个单元。如果我们假设 $S=z$，我们获得 CNN 中传统的局部池化。如果设 $S<z$,我们获得重叠池化。

三 AlexNet 网络架构

为了更好地讲解卷积操作，我们首先的对卷积操作和图像尺寸之间的关系进行说明。假设图像维度为 $W\times W\times L$，即图像由 L 个 $W\times W$ 的二维矩阵构成。卷积内核维度为：$F\times F\times L\times K$，即由 K 个 $F\times F\times L$ 三维矩阵构成。步长为 S，使用了 P 次全 0 填充。那么经过卷积操作后，内核映射的尺寸为：$\frac{W-F-2P}{S}+1$。

接下来我们将逐层介绍 AlexNet 的网络架构。图 3.1 是 AlexNet 的整体架构图。

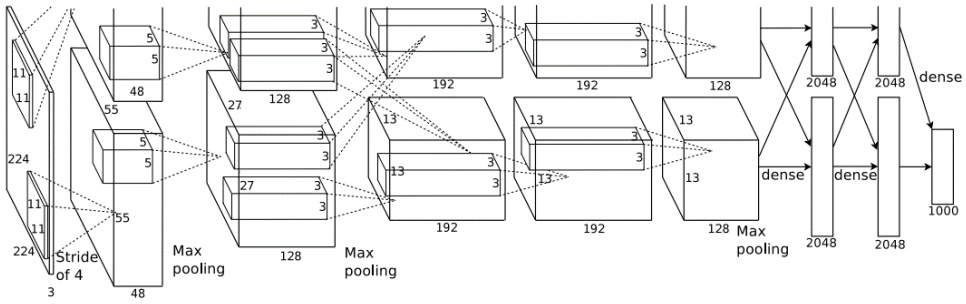

图 3.1　AlexNet 架构图

3.1　卷积层 1(Conv1)

输入维度为：$224\times224\times3$，即这也是 ALexNet 的输入的起点。内核维度为：$11\times11\times3\times96$，即由尺寸为 11，3 个通道的滤波器(掩膜、模板)与输入的原始图像进行卷积操作。步幅为 4，即卷积核沿原始图像的 x 轴方向和 y 轴方向两个方向移动，移动的步长是 4 个像素。该层未做全 0 填充。**注明：根据上述图中给出的 AlexNet 网络架构，卷积之后的结果维度为 $55\times55\times96$，那么倒退回去输入图像维度应该为 $227\times227\times3$，那么我们必须视作输入图像输入到 AlexNet 之后进行了预处理，将原来的 $224\times224\times3$ 扩展到了 $227\times227\times3$。在下面的笔记中我们主要是以 $227\times227\times3$ 为输入进行讲述。**那么经过卷积操作之后的图像尺寸为：$\frac{227-11}{4}+1=55$。那么卷积之后的图像维度为：$55\times55\times96$。之后这 $55\times55\times96$ 的卷积结果被随机分成了两组，即 2 个 $55\times55\times48$，之后在 2 个 GPU 上分别利用 ReLU 非线性激活函数激活后送至最大池化层。Conv1 紧跟着的最大池化层的内核维度为：3×3，步幅为 2，那么输入图像的尺寸为：$\frac{55-3}{2}+1=27$，即输出结果为 2 个维度为 $27\times27\times48$ 的图像，之后这个组图像进行合并为：$27\times27\times96$。在最大池化层操作之后，该层还进行了**局部响应归一化(LRN)**。

3.2　卷积层 2(Conv2)

输入维度为：2 个 $27\times27\times48$ (统一看成是 $27\times27\times96$)，卷积内核尺寸为：2 个 $5\times5\times48\times256$ (或者统一看成是 $5\times5\times96\times256$)。步幅为 1，在该层进行了全 0 填充，上下左右各填充 2 个像素。那么经历卷积操作之后的图像尺寸为：$\frac{27-5+2\times2}{1}+1=27$，即 2 块 GPU 上的图像输入经过卷积之后的图像维度均为：$27\times27\times128$。然后将卷积之后的结果通过 ReLU 函数进行激活后送至最大池化层。其中，内核维度为：3×3，步幅为 2。那么 2 块 GPU 上经过最大池化层得到的图像尺寸为：$\frac{27-3}{2}+1=13$，即图像维度为：$13\times13\times128$。与 Conv1 层一样，最大池化层的处理后的结果在也进行了 LRN。之后将 2 块 GPU 上的结果进行合并，那么图像维度为：$13\times13\times256$。

3.3 卷积层 3(Conv3)

输入维度为：13×13×256，卷积内核尺寸为：3×3×256×384。步幅为 1，在该层进行了全 0 填充，上下左右各填充 1 个像素。那么经历卷积操作之后的图像尺寸为：$\frac{13-3+2\times1}{1}+1=13$，图像输入经过卷积之后的图像维度为：13×13×384。然后将卷积之后的结果通过 ReLU 函数进行激活，之后还进行了 LRN。

3.4 卷积层 4(Conv4)

输入维度为：13×13×384，卷积内核尺寸为：3×3×384×384。步幅为 1，在该层进行了全 0 填充，上下左右各填充 1 个像素。那么经历卷积操作之后的图像尺寸为：$\frac{13-3+2\times1}{1}+1=13$，图像输入经过卷积之后的图像维度为：13×13×384。然后将卷积之后的结果通过 ReLU 函数进行激活，之后还进行了 LRN。之后将处理得到的结果平分成 2 组送至 2 个 GPU，即平分后的图像维度均为13×13×192。与 Conv1 层一样，最大池化层的处理后的结果在也进行了 LRN。

3.5 卷积层 5(Conv5)

输入维度为：2 个13×13×192，卷积内核尺寸为：2 个3×3×192×128。步幅为 1，在该层进行了全 0 填充，上下左右各填充 1 个像素。那么经历卷积操作之后的图像尺寸为：$\frac{13-3+2\times1}{1}+1=13$，图像输入经过卷积之后的图像维度为：2 个13×13×128。然后将卷积之后的结果通过 ReLU 函数进行激活。之后进入最大池化层操作，内核维度为：3×3，步幅为：2。那么经过最大池化图像尺寸为：$\frac{13-3}{2}+1=6$，即该层图像维度为 2 个6×6×128，并做了 LRN。之后将 2 个 GPU 处理得到结果合并，即图像维度变成：6×6×256。与 Conv1 层一样，最大池化层的处理后的结果在也进行了 LRN。

3.6 全连接层 1(FC1)

在此之后的网络结果可以看成是简单的前馈神经网络。其输入维度为：6×6×256，为了进行前向传播，必须将输入图像矩阵拉成一维向量，即转化为：$1\times(6*6*256)=1\times9216$。Conv5 层与 FC1 层之间权重的维度为：9216×4096，偏置的维度为：1×4096。前行传播的结果的维度为：1×4096。激活函数运用的是 ReLU 函数。同时将 FC1 和 FC2 两层之间的权重做了 Dropout 处理，概率为 0.5。

3.7 全连接层 2(FC2)

输入维度为：1×4096，FC2 层与 FC3 层之间权重的维度为：4096×4096，偏置的维度为：1×4096。前行传播的结果的维度为：1×4096。激活函数运用的是 ReLU 函数。同时将 FC1 和 FC2 两层之间的权重做了 Dropout 处理，概率为 0.5。

3.8 全连接层 3(FC3)

输入维度为：1×4096，FC2 层与 FC3 层之间权重的维度为：4096×1000，偏置的维度为：1×1000。前行传播的结果的维度为：1×1000。激活函数运用的是 Softmax 函数。得到的 1000 维向量代表了 1000 中分类对应的概率。

四 避免过拟合

首先来说一下，AlexNet 用到的数据集——LSVRC2010。该数据集是 ImageNet 的子集。ImageNet 数据集包含有大概 22000 种类别共 150 多万带标签的高分辨率图像。这些图像是从网络上收集得来，由亚马逊的 Mechanical Turkey 的众包工具进行人工标记。从 2010 年开始，作为 Pascal 视觉目标挑战的一部分，ImageNet 大规模视觉识别挑战（ImageNet Large-Scale Visual Recognition Challenge ，ILSVRC）比赛每年都会举行。

ILSVRC 采用 ImageNet 的子集，共包含一千个类别，每个类别包含大约 1000 幅图像。总的来说，大约有 120 万张训练图像，5 万张验证图像以及 15 万张测试图像。ILSVRC-2010 是 ILSVRC 唯一一个测试集标签公开的版本，因此这个版本就是本文大部分实验采用的数据集。ImageNet 通常使用两种错误率：top-1 和 top-5，其中 top-5 错误率是指正确标签不在模型认为最有可能的前五个标签中的测试图像的百分数。

这个数据集图像分辨率为 256×256。论文中对图像的处理为：图像本身减去训练集的均值图像(训练集和测试集都减去训练集的均值图像)。因此论文直接在每个像素的原始 RGB 值上进行训练。

4.1 数据扩展

降低图像数据过拟合的最简单常见的方法就是利用标签转换人为地增大数据集。本文采取两种不同的数据增强方式，这两种方式只需要少量的计算就可以从原图中产生转换图像，因此转换图像不需要存入磁盘。本文中利用 GPU 训练先前一批图像的同时，使用 CPU 运行 Python 代码生成转换图像。因此这些数据增强方法实际上是不用消耗计算资源的。

第一种数据增强的形式包括生成平移图像和水平翻转图像。做法就是从 256×256 的图像中提取随机的 224×224 大小的块（以及它们的水平翻转），然后基于这些提取的块训练网络。这个让我们的训练集增大了 2048 倍($(256-224)^2\times2=2028$)，尽管产生的这些训练样本显然是高度相互依赖的。在测试时，网络通过提取 5 个 224×224 块（四个边角块和一个中心块）以及它们的水平翻转（因此共十个块）做预测，然后网络的 softmax 层对这十个块做

出的预测取均值。

第二种数据增强的形式包括改变训练图像的 RGB 通道的强度。特别的，本文对整个 ImageNet 训练集的 RGB 像素值进行了 PCA。对每一幅训练图像，本文加上多倍的主成分，倍数的值为相应的特征值乘以一个均值为 0 标准差为 0.1 的高斯函数产生的随机变量。因此对每一个 RGB 图像像素 $I_{xy}=\left[I_{xy}^{R},I_{xy}^{G},I_{xy}^{B}\right]^{T}$ 都将加上了 $\left[p_1,p_2,p_2\right]\left[\alpha_1\lambda_1,\alpha_2\lambda_2,\alpha_3\lambda_3\right]^{T}$

这里 p_i 和 λ_i 是 RGB 像素值的 3×3 协方差矩阵的第 i 个特征向量和特征值，α_i 是上述的随机变量。每一个 α_i 的值对一幅特定的训练图像的所有像素是不变的，直到这幅图像再次用于训练，此时才又赋予 α_i 新的值。这个方案得到了自然图像的一个重要的性质，也就是，改变光照的颜色和强度，目标的特性是不变的。

4.2 Dropout

dropout 它将每一个隐藏神经元的输出以 0.5 概率设为 0。以这种方式被“踢出”的神经元不会参加前向传递，也不会加入反向传播。因此每次有输入时，神经网络采样一个不同的结构，但是所有这些结构都共享权值。这个技术降低了神经元之间复杂的联合适应性，因为一个神经元不是依赖于另一个特定的神经元的存在的。因此迫使要学到在连接其他神经元的多个不同随机子集的时候更鲁棒性的特征。在测试时，本文使用所有的神经元，但对其输出都乘以了 0.5。

五 学习细节

训练过程中，使用 SGD 进行优化模型参数，小样本规模为 128，动量因子为 0.9，权重衰减(L2 正则化)系数为 0.0005。论文发现这个很小的权值衰减对模型的学习很重要。换句话说，这里的权值衰减不只是一个正则化矩阵：它降低了模型的训练错误率。权重 w 更新规则是：

$$v_{i+1}=0.9v_i\text{-}0.0005\varepsilon w_i-\varepsilon\left\langle\frac{\partial L}{\partial w}|_{w_i}\right\rangle_{D_i}$$

$$w_{i+1}=w_i+v_{i+1}$$

其中，i 为迭代次数，v 为动量变量，ε 为学习率。$\left\langle\frac{\partial L}{\partial w}|_{w_i}\right\rangle_{D_i}$ 是第 i 的小样本 D_i 上，损失函数 L 对变量 w 平均梯度。

对于权重，整个 AlexNet 中所有权重全部设置成均值为 0，标准差为 0.01 的高斯分布随机数。对于偏置，第 2、4 和 5 卷积层和全连接层的偏置初始化为 1.0，其余层的偏置初始化为 0。

学习率使用固定学习率 0.01，当验证误差停止降低时，学习率除以 10。